Introduction of AI (CS103)- 12 Deep Learning - BP

Jimmy Liu 刘江

2023-12-08

Lecture 12

1. Reviews of Lecture 11
2. Deep Learning-Introduction
3. Deep Learning- Back Propagation
4. Deep Learning Models

CS103 Project Presentation Requirements

- Held in weeks 14-16. The specific arrangements are shown on the next page.
- Each group is asked to record an 8-10 minutes defense video for playback in class. There will be a Q&A session after t. For all groups, please submit your defense video to your TA at least two days (Wednesday) before the defense.
- Review teams should submit their review papers before the weekend of the 13th week (23:59 on December 17, 2023);
- Project teams should submit their project file together with a technical report before the weekend of the 16th week (23:59 on January 7, 2024). The technical report should briefly explain the code composition, operation method, effect display, etc. of your project.

CS103 Presentation Schedule

	Week 14		Week 15		Week 16	
	2023.12.22		2023.12.29		2024.1.5	
	Group Num	Leader	Group Num	Leader	Group Num	Leader
10:20 ~ 11:10	7	杨祎勃	3	朱家润	16	杨烜
	8	吴成钢	2	陶文晖	5	谢毅
	9	曾宪清	12	冯泽欣	1	虞快
	10	蒋浩天	18	徐进哲	15	王德涵
	14	罗雨涵				
10min break						
11:20 ~ 12:10	17	杨宗奇	13	黄宇航	**Brief review of this course**	
	19	李轲	6	李岩		
	20	TANG RYAN TZE HOU	4	陈旭阳		
	21	陈茜	11	李博洋		

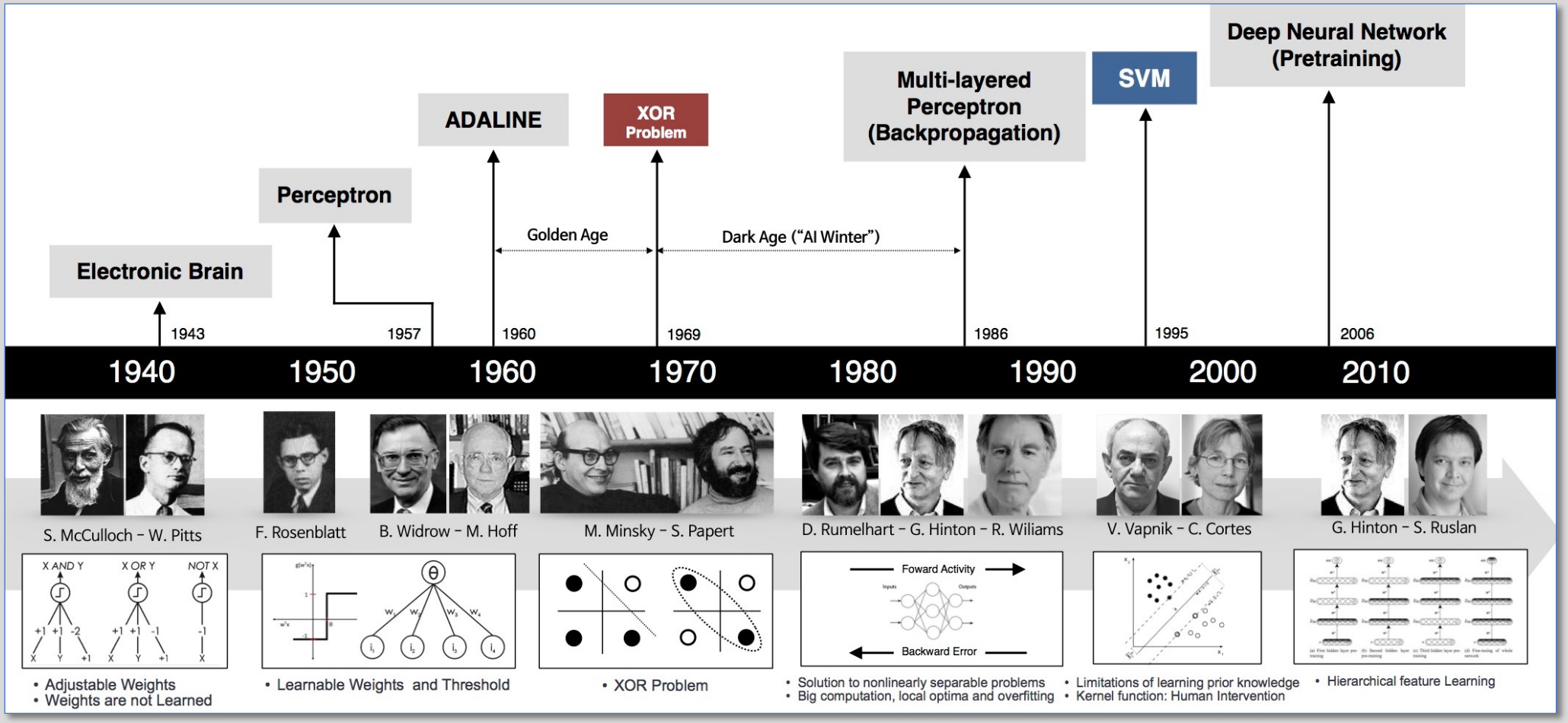
EXAM Time Table

CS103	人工智能导论	计算机科学与工程系	刘江	1-16	第16周 星期五	第18周 星期五	2024-01-19	10:30-12:30	期末考试	三教106	85	80
CS103	人工智能导论	计算机科学与工程系	刘江	1-16	第16周 星期五	第18周 星期五	2024-01-19	10:30-12:30	期末考试	三教307	30	26

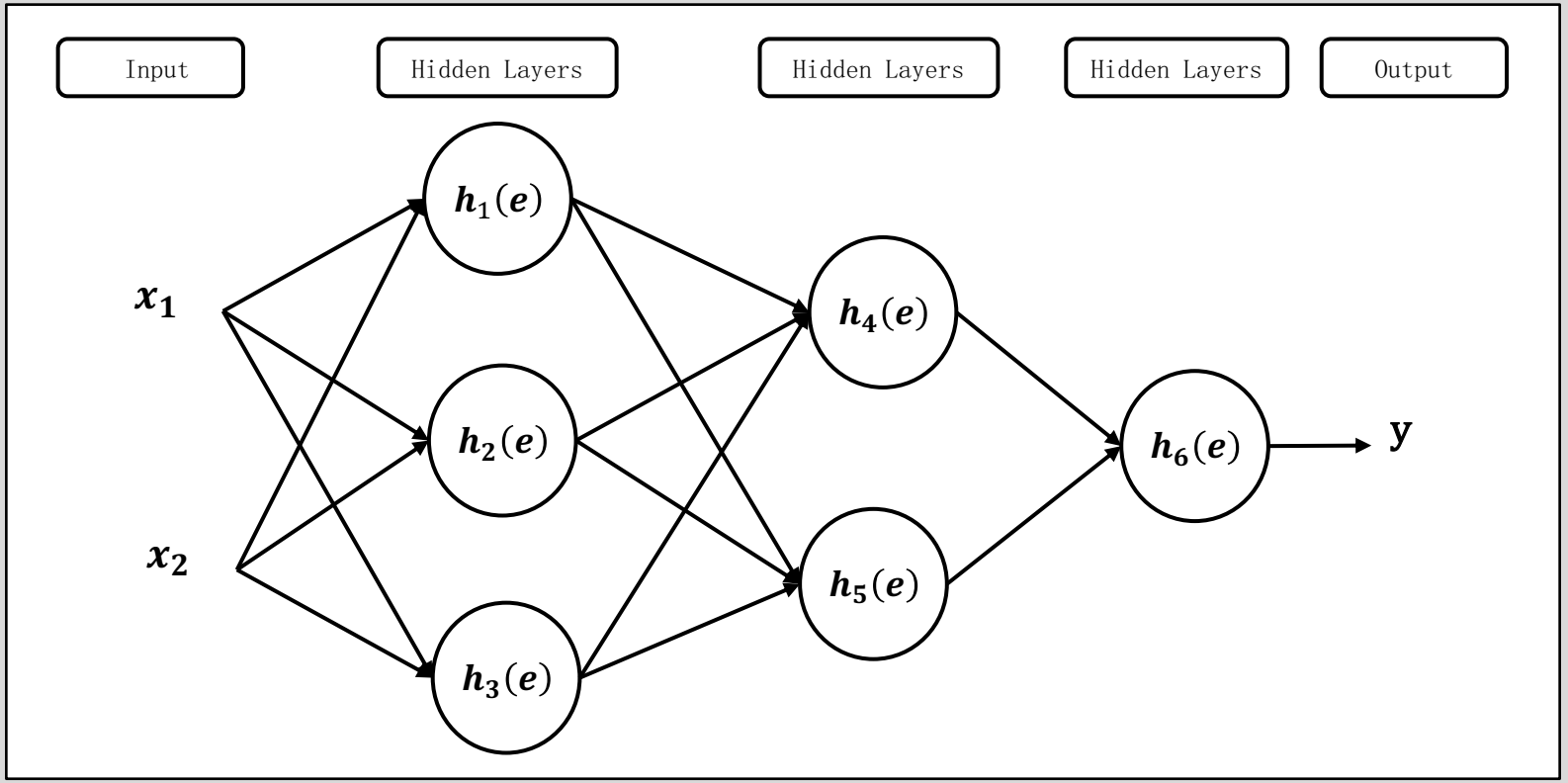
Lecture 12

1. Reviews of Lecture 11
2. Deep Learning-Introduction
3. Deep Learning- Back Propagation
4. Deep Learning Models

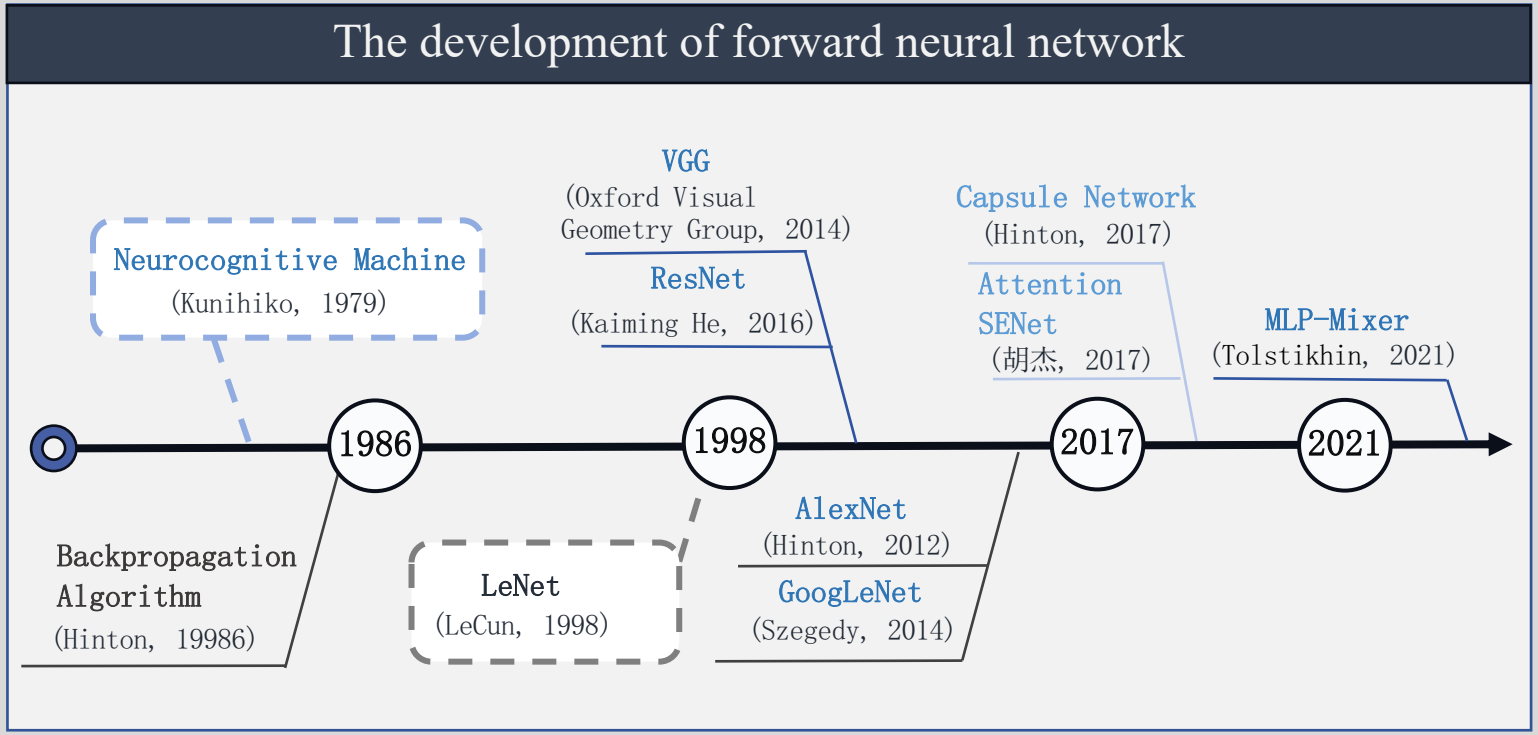
AI Algorithm Development



Multi-layer Neural Network



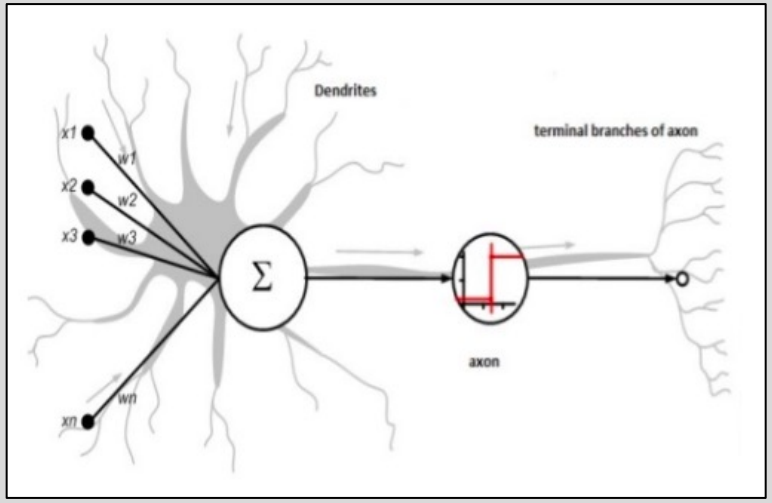
Forward Neural Network Model



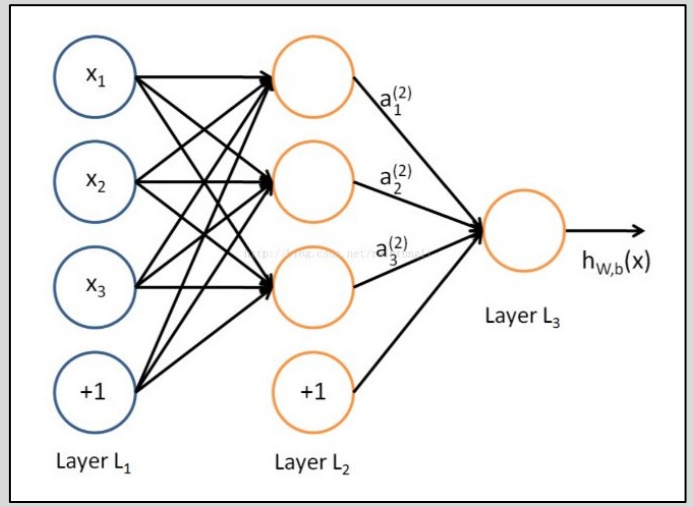
Forward Neural Network Model

Japanese scholar ：Kunihiko Fukushima

Forward neural network, this term comes from the Neocognitron proposed by Japanese scholar Kunihiko Fukushima in 1979. In addition, the ideas of convolution and pooling were also given for the first time in the Neurocognitron.



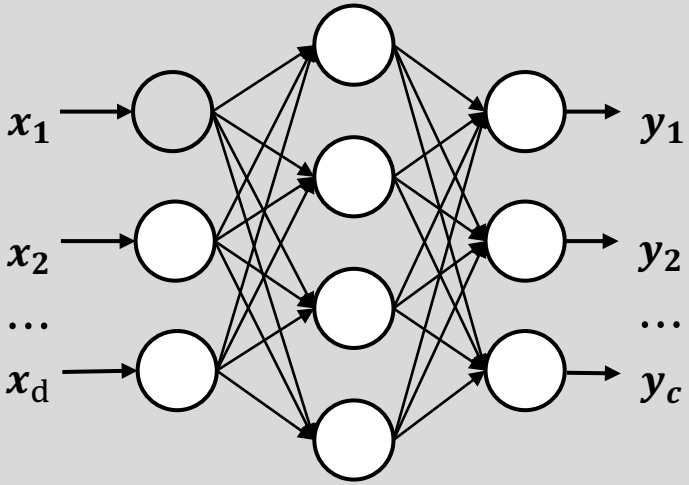
neurocognitive machine



forward neural network

Forward Neural Network Model

Geoffrey Hinton



In 1986, Geoffrey Hinton proposed to use the backpropagation algorithm to train a multi-layer perceptron, and solved the problem that the perceptron could not handle nonlinear learning.

BP Algorithm

The backpropagation algorithm was proposed to solve the learning problem of multi-layer neural networks, and it has become the most popular learning algorithm for neural network models.

There are three core differences between the update rules of BP and perceptron:

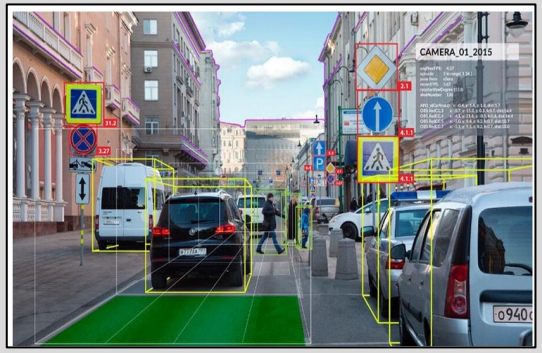
- Utilize nonlinear activation of BP hidden units;
- The BP rule contains the gradient term of the activation function and uses gradient descent to minimize the error;
- BP cannot use perceptron learning rules because hidden units cannot have teacher values.

Deep Learning

Turing Award Winner: Geoffrey Hinton

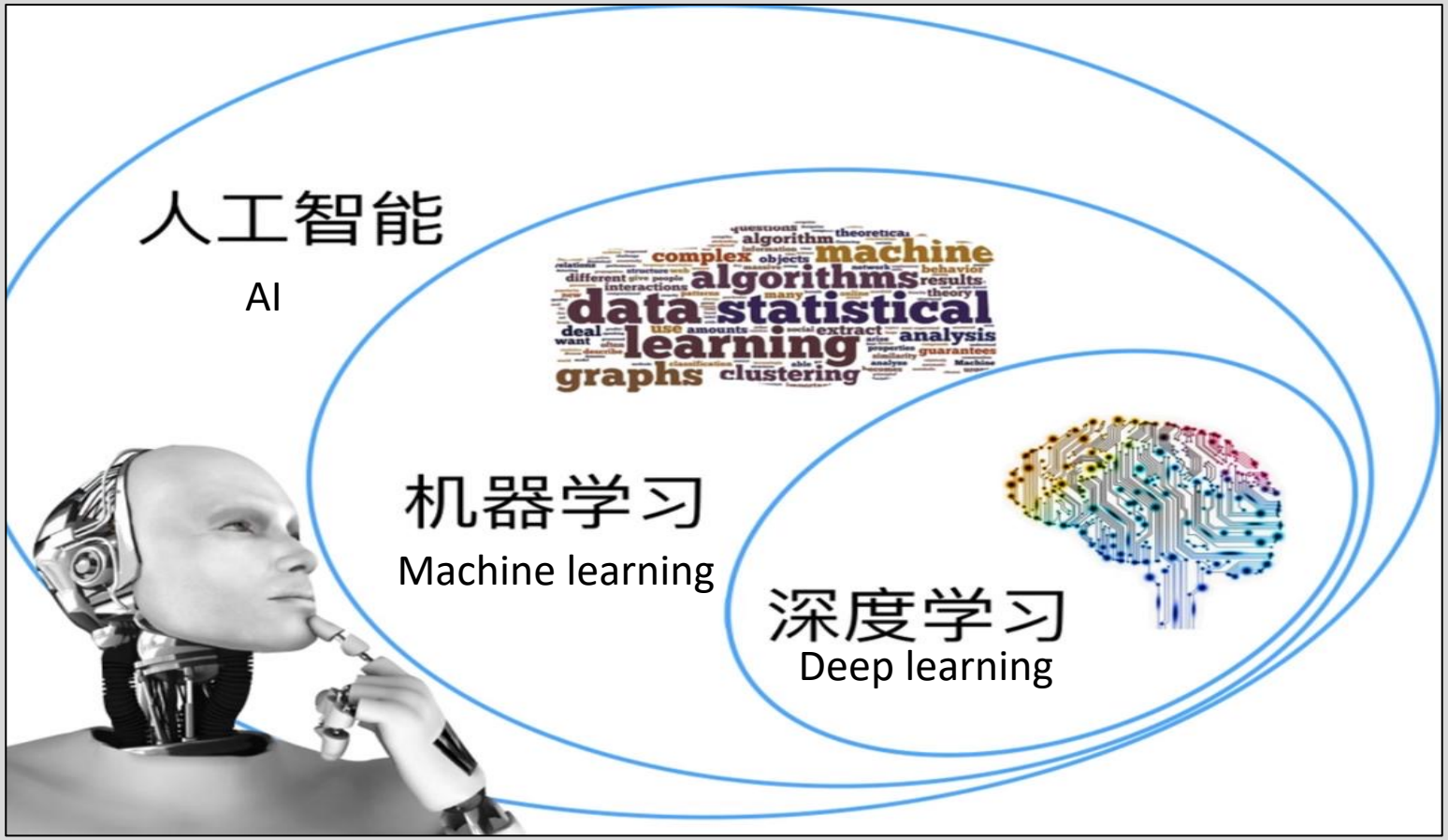
In 2006, Hinton published a paper on the neural network deep learning algorithm in Science, which improved the information storage capacity of the neural network, also known as network capacity. This research has greatly promoted the development of deep learning research, making deep learning widely used in **computer vision**, **natural language processing**, **smart medical care**, **smart cities** and other fields.

Deep Learning



The increase in network capacity has promoted the development of deep learning research, making deep learning widely used in fields such as **computer vision**, **natural language processing**, **smart medical care**, and **smart cities**.

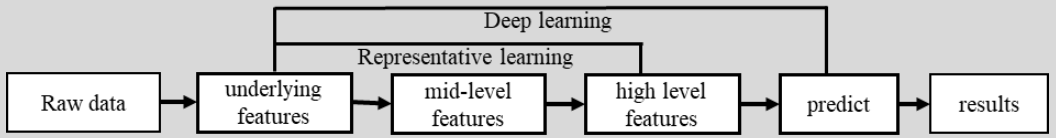
Deep Learning and AI



Deep Learning

- Deep learning is a sub-research direction of machine learning. Its main goal is to automatically learn effective feature representations from data by building a certain "depth" model, overcome the shortcomings of traditional machine learning that relies too much on feature engineering, and achieve end-to-end learning.
- "Depth" is mainly reflected in the number of nonlinear feature transformations performed on the original data. If the deep learning model is regarded as a tree structure, the depth can be regarded as the length of the maximum path from the root node to the leaf node.
- End-to-end learning refers to the overall goal of directly optimizing the task without performing sub-module or phased training in the learning process.

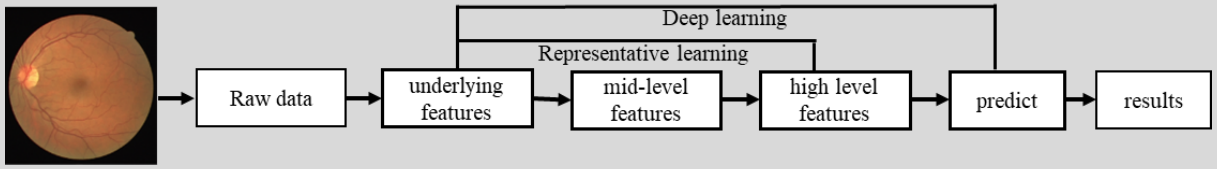
Deep Learning



Data processing flow chart of deep learning model

The figure shows the data processing flow chart of a deep learning model, which converts the original data into a higher-level, more abstract feature representation through multiple nonlinear transformations, and further inputs it into the prediction function to generate the final result. Compared with the "shallow learning" of traditional machine learning, the key technical problem that deep learning needs to solve is the **credit assignment problem** (CAP), that is, the contribution of different sub-components or parameters in a model to the final output of the model. contribution or influence.

Deep Learning



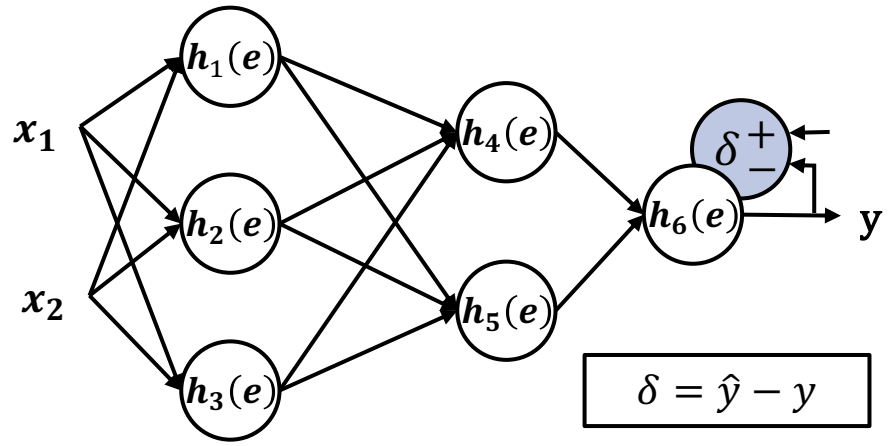
Deep learning model for glaucoma diagnosis

Taking the diagnosis of glaucoma disease as an example, we usually use fundus as input. Each component in the deep neural network will process the image information to generate different types of feature representations: low-level, mid-level and high-level feature representation. The low-level feature representation, as the input of the mid-level feature representation generation component, will have an impact or contribution on the type of mid-level feature representation that the component outputs. By analogy, it is difficult for us to clearly judge how each component contributes to the accurate glaucoma output of the neural network model. What is the contribution of disease prediction results or which components influence the neural network model to output incorrect glaucoma disease prediction results.

Lecture 12

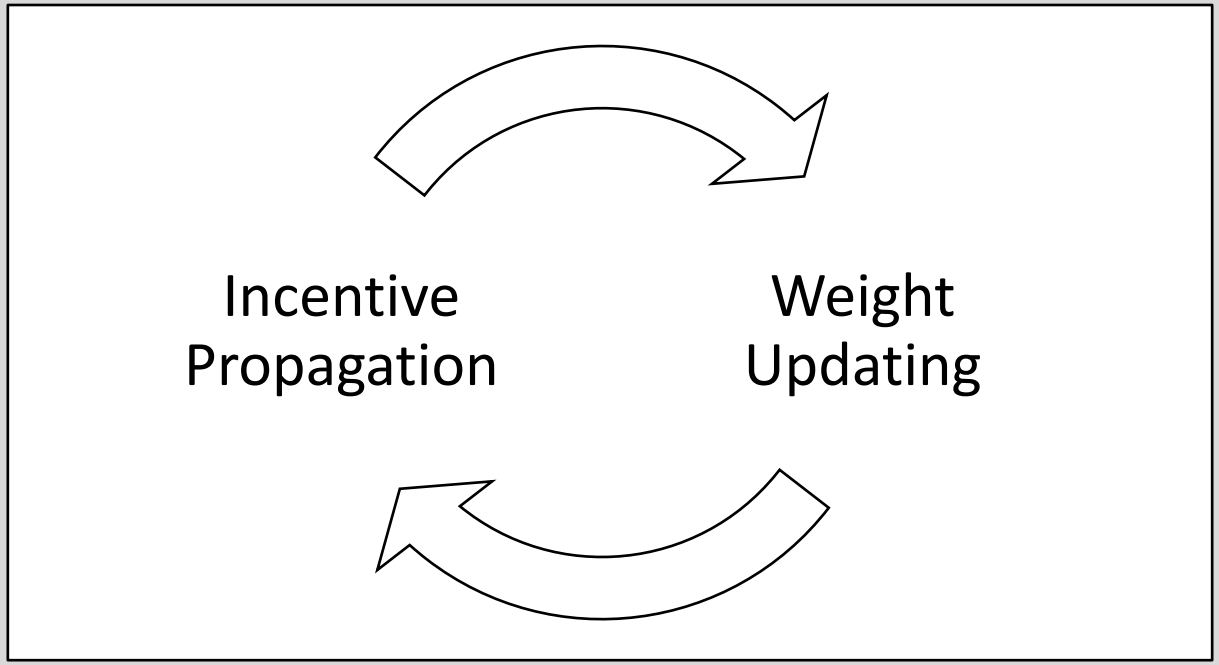
1. Reviews of Lecture 11
2. Deep Learning-Introduction
3. Deep Learning- Back Propagation
4. Deep Learning Models

Backpropagation Algorithm

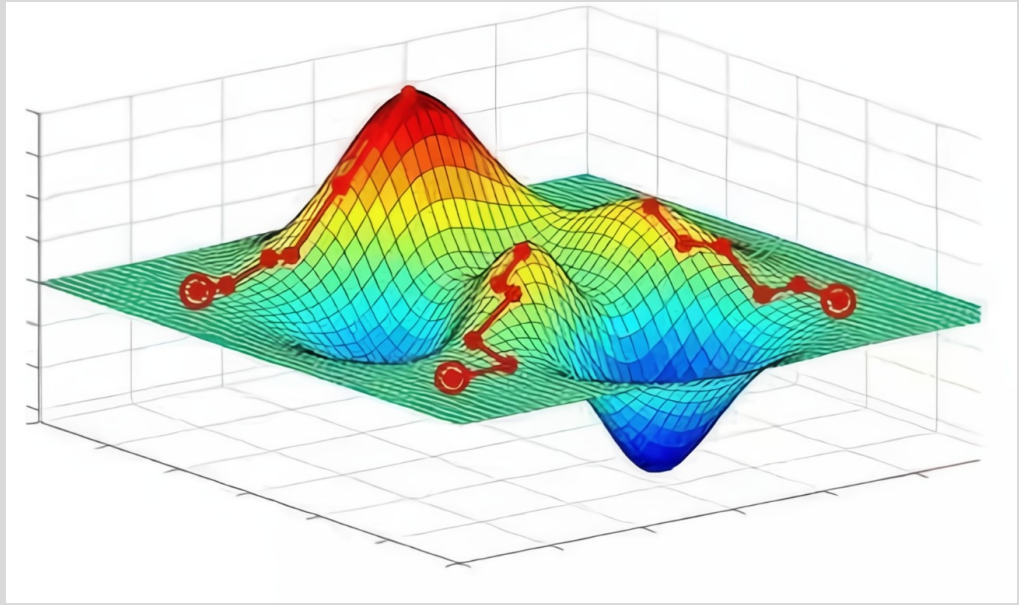


Back Propagation (BP) algorithm is an algorithm used for gradient propagation in multilayer neural networks. It is the method used to compute gradients. In essence, it calculates the gradient of the loss function with respect to the weights at each layer of the network using the chain rule and then updates the weights to minimize the loss function.

Backpropagation Algorithm



Parameter Optimization: Gradient Descent



Gradient descent is a method for minimizing the objective function and an iterative algorithm that utilizes gradient information to adjust parameters through continuous iterations to find an optimal solution.

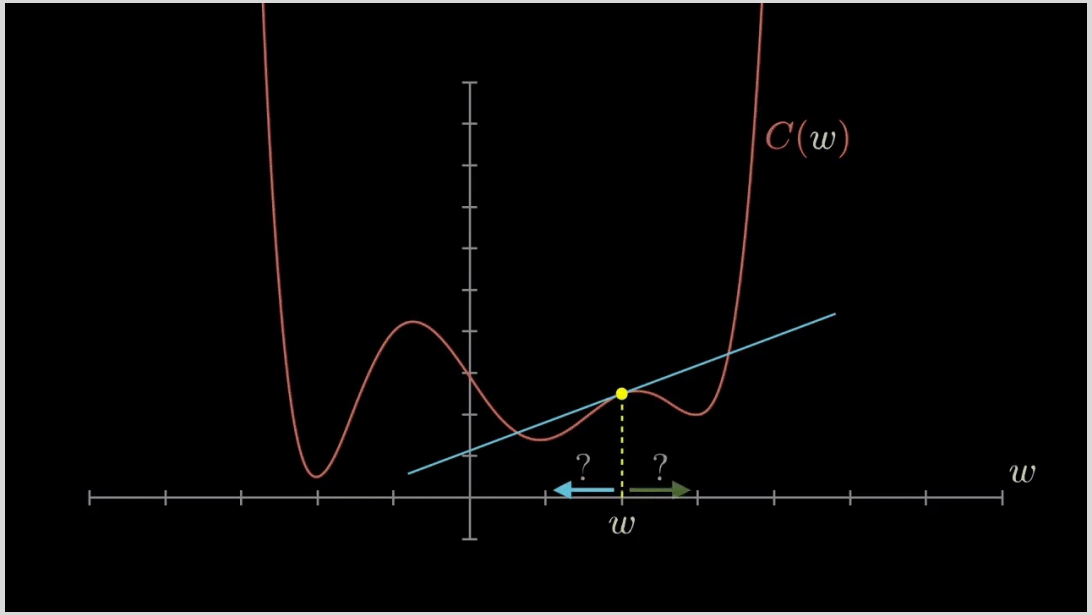
Gradient

- The gradient of the function f composed of a single variable at x is:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

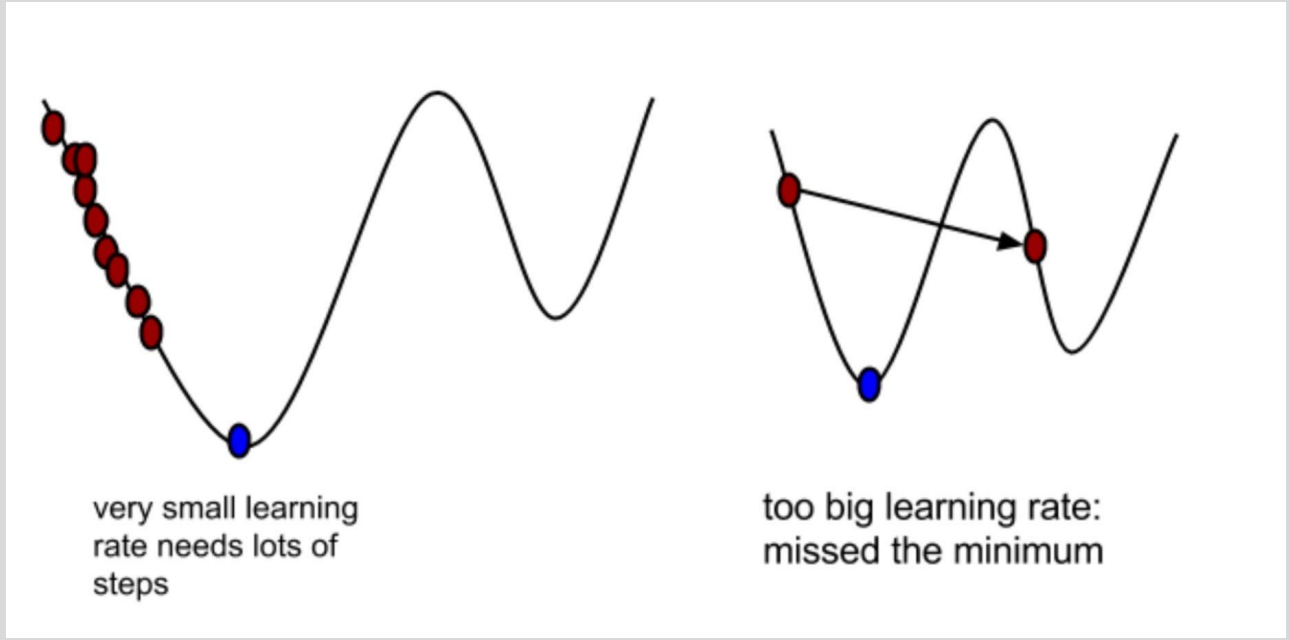
- In multivariate functions, the gradient is a vector composed of derivatives with respect to each variable.

Gradient



From a mathematical perspective, the positive direction of the gradient is the direction in which the function grows most rapidly, so the negative direction of the gradient is the direction in which the function decreases most rapidly.

Gradient Descent Method



Gradient Descent Method

Assuming $x^{(k)}$ represents the result of the k-th iteration of the gradient descent method, perform a first-order Taylor expansion of the function $f(x)$ around $x^{(k)}$:

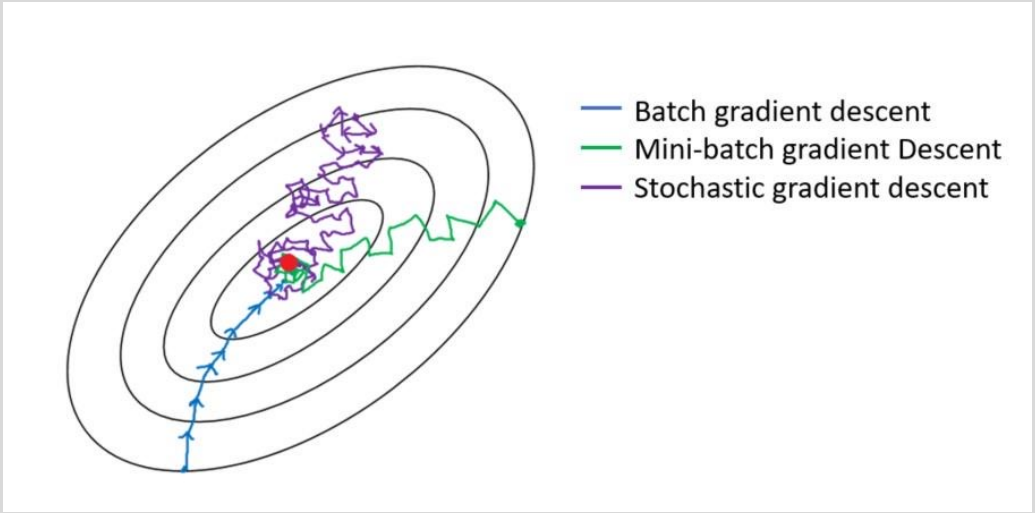
$$f(x) = f\left(x^{(\mathrm{k})}\right) + \nabla f\left(x^{(\mathrm{k})}\right)\left(x - x^{(\mathrm{k})}\right)$$

In which $\nabla f\left(x^{(k)}\right)$ represents the gradient of the function $f(x)$ at $x^{(k)}$.

The update formula for the result $x^{(k+1)}$ of the $k+1$-th iteration of the gradient descent method is as follows:

$$x^{(\mathrm{k}+1)} = x^{(\mathrm{k})} - \lambda_k \nabla f\left(x^{(\mathrm{k})}\right)$$

Gradient Descent Method



Gradient descent method can be divided into:

- Batch Gradient Descent
- Stochastic Gradient Descent
- Mini-Batch Gradient Descent

Stochastic Gradient Descent

Algorithm 1 Stochastic Gradient Descent (SGD) Algorithm

Input: training set $D = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$, initial learning rate α_0

Output: model parameter θ

1: randomly shuffle the training set samples

2: set learning rate $\alpha \leftarrow \alpha_0$

3: **repeat**

4: **for** $n = 1 \ldots$ N **do**

5: select sample $(x^{(n)}, y^{(n)})$ from the training set D

6: $\theta \leftarrow \theta - \alpha \frac{\partial f(\theta; x^{(n)}, y^{(n)})}{\partial \theta}$

7: **end for**

8: update learning rate α

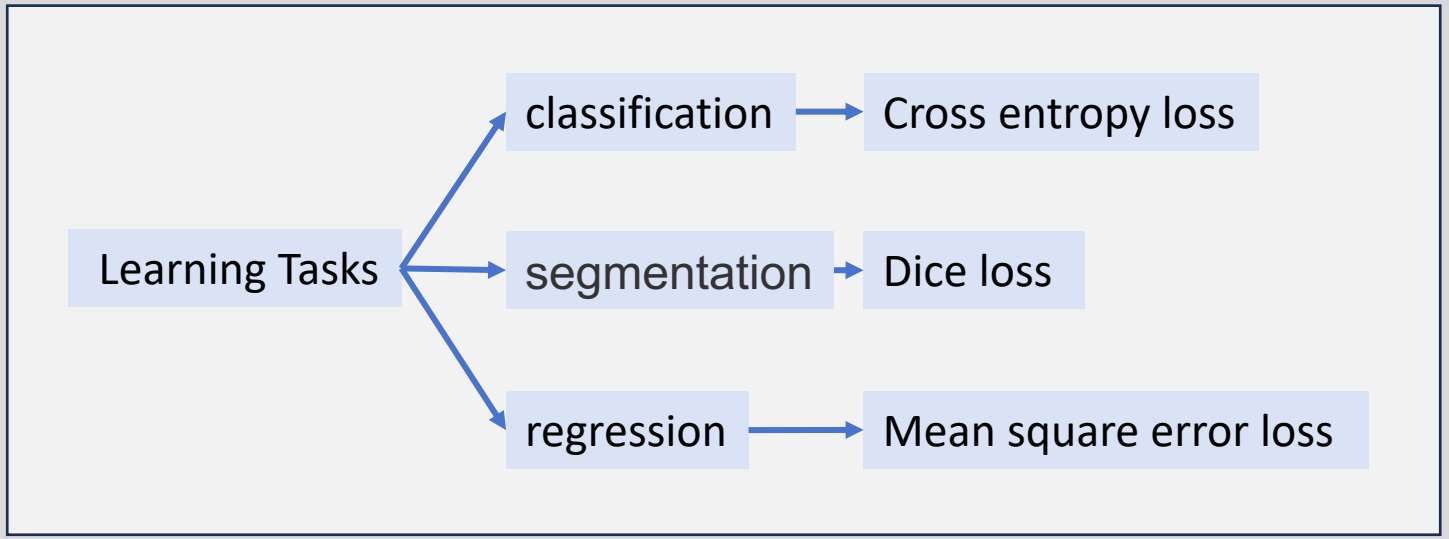
9: **until** the target value of model optimization converges or reaches the set number of times

Loss Function

Frank Rosenblatt

The loss function (or cost function) is a function used in neural networks to measure the difference between predicted and actual values. Its earliest use can be traced back to the loss function already used in the Perceptron model proposed by Rosenblatt in 1958, known as the "Perceptron Criterion".

Loss Function



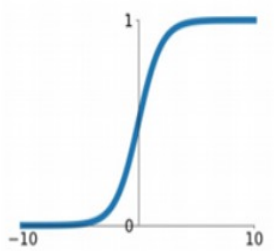
The loss function is an important component of a neural network model, which needs to be selected or designed according to the task. During the training process, the loss function is used to calculate the error size of the neural network, and then gradient descent and backpropagation algorithms are used to optimize the parameters in the neural network.

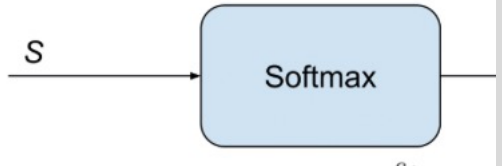
Classification: Cross entropy loss

For classification tasks, whether it is binary or multi classification, neural network models usually use the Sigmoid function or Softmax function, whose output is the probability values predicted by the samples for each category. The output sample prediction value of the machine learning algorithm is probability; In multi classification problems, the predicted values of the Softmax function output samples in the neural network are also probabilities.

Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$





$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}}$$

Classification: Cross entropy loss

- We generally use the classical cross entropy (CE) loss function to calculate the error between the predicted values of the model and the actual label values. Cross entropy is an important concept in information theory, which is mainly used to measure the differential information between two probability distributions. Given a sample x, p and q are their two conceptual distributions, the cross entropy of p is represented by q, and the formula is as follows：

$$H(p,q) = -p(x)\,log\,q(x)$$

- Cross entropy characterizes the distance between two probability distributions, aiming to describe the difficulty of expressing probability p through probability q. Based on the above formula, it can be seen that the smaller the cross entropy value, the closer the probability values of p and q are.

Example of Taking glaucoma binary classification detection

Taking glaucoma detection (binary classification) as an example: given 4 fundus images of glaucoma, the labels are {1,0,1,0}, where 1 is the fundus image with glaucoma and 0 is the normal fundus image. The output probability values of the neural network model are {0.6,0.6,0.3,0.4}, respectively. There are only two scenarios for the prediction results of binary classification. For each category, the probabilities of the predicted output are p and 1-p. Therefore, the expression of the cross entropy loss function is as follows:

$$L = \frac{1}{n}\sum_{i=1}^{n} -[y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)],$$

where n represents the sample size， y_i represents the label of the i-th sample， p_i indicates the probability of predicting the i-th sample as a positive class.

Example of Taking glaucoma binary classification detection

Therefore, for the binary classification example of glaucoma detection, we can calculate the cross entropy loss of the sample as follows:

$$L = -\frac{1}{4}[(1\times\log 0.6 + 0\times\log(1-0.6)) + (0\times\log 0.6 + 1\times\log(1-0.4))] +$$
$$(1\times\log 0.3 + 0\times\log(1-0.3)) + (0\times\log 0.4 + 1\times\log(1-0.4))$$
$$= -\frac{1}{4}(\log 0.6 + \log 0.6 + \log 0.3 + \log 0.6) \approx 0.6841$$

Multi classification: Cross entropy loss function

For multi classification problems, assume there are n samples in the training dataset, and the true number of label categories in the samples is c. For the nth sample, its true label is a one-dimensional vector $\boldsymbol{y}^{(i)} \epsilon \mathbb{R}^c$. When the sample belongs to class j, the nth element of the sample is $y_j^{(i)}$ is 1, and the rest is 0. Here we use $\hat{\boldsymbol{y}}^{(i)}$ represents the predicted result. Therefore, the form of its cross entropy is as follows:

$$CrossEntropy\left(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)}\right) = -\sum_{j=1}^{c} y_j^{(i)} log \hat{y}_j^{(i)}$$

Due to the element $y_j^{(i)}$ value is only 0 or 1. From the representation of cross entropy, it can be seen that cross entropy only focuses on the prediction probability of the true category of the sample. For a dataset with n samples, the cross entropy loss function can be expressed as

$$L = \frac{1}{n} \sum_{i=1}^{n} CrossEntropy\left(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)}\right)$$

Example of Taking the Three Classification and Grading of Cataracts

Assuming there are three fundus images of cataracts with labels {0, 1, 2}, where 0 represents a normal sample, 1 represents a mild cataract patient, and 2 represents a severe cataract patient. In these three classification problems, we first use one hot encoding technology to convert the true labels of each sample into a three-dimensional vector, and the prediction result is also a three-dimensional vector. The probability of each one-dimensional corresponding sample being predicted as the corresponding category is calculated. The sample information is shown in the table below:

Table Real labels and predicted results of cataract samples

number	Real label	Predict result
1	(1,0,0)	(0.6,0.2,0.2)
2	(0,1,0)	(0.1,0.7,0.2)
3	(0,0,1)	(0.3,0.3,0.4)

Example of Taking the Three Classification and Grading of Cataracts

Here, we first calculate the cross entropy of each sample using the following formula:

$$CrossEntropy\left(\boldsymbol{y}^{(1)}, \hat{\boldsymbol{y}}^{(1)}\right) = -[1\times \log 0.6 + 0\times \log 0.2 + 0\times \log 0.2] \approx 0.5108$$
$$CrossEntropy\left(\boldsymbol{y}^{(2)}, \hat{\boldsymbol{y}}^{(2)}\right) = -[0\times \log 0.1 + 1\times \log 0.7 + 0\times \log 0.2] \approx 0.3567$$
$$CrossEntropy\left(\boldsymbol{y}^{(3)}, \hat{\boldsymbol{y}}^{(3)}\right) = -[0\times \log 0.3 + 0\times \log 0.3 + 1\times \log 0.4] \approx 0.9163$$

According to the definition of the cross entropy loss function, calculate the cross entropy loss for all sample categories as follows：

$$L = \frac{1}{3}\sum_{i=1}^{3} CrossEntropy\left(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)}\right) = \frac{0.5108 + 0.3567 + 0.9163}{3} \approx 0.5946$$

Real labels and predicted results of cataract samples

number	Real label	Predict result
1	(1,0,0)	(0.6,0.2,0.2)
2	(0,1,0)	(0.1,0.7,0.2)
3	(0,0,1)	(0.3,0.3,0.4)

Segmentation: Dice Loss Function

The Dice loss function is a commonly used loss function in image segmentation tasks, suitable for cases of extremely uneven samples, and therefore is often used for medical image segmentation. It is designed on the Dice coefficient of the set similarity measurement function:

$$Dice\ Coefficient = \frac{2|X \cap Y|}{|X| + |Y|},$$

where $|X \cap Y|$ is the intersection of X and Y, $|X|$ and $|Y|$ represent the number of elements in X and Y, respectively. Due to the repeated calculation of the common elements of X and Yin the denominator $|X| + |Y|$, the coefficient of the numerator $|X \cap Y|$ is 2.

Example of Segmenting Fundus in OCTA Images

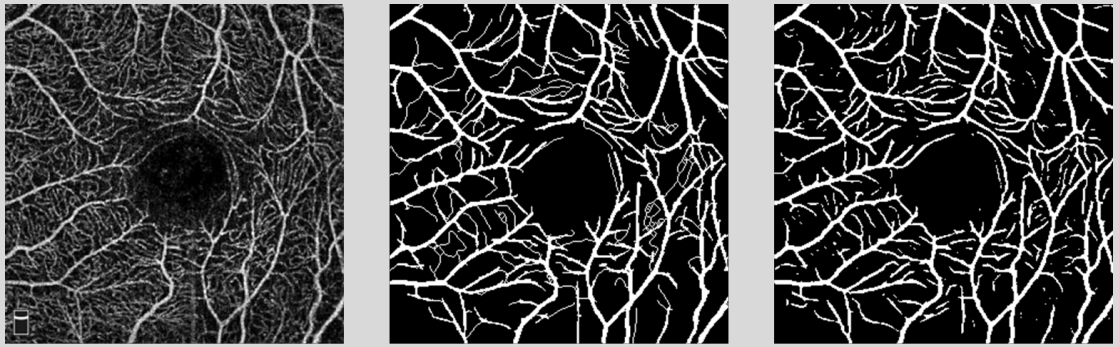
The first image on the left is of the retinal superficial vascular complex, with the segmentation label in the middle and the segmentation prediction results generated by the neural network on the far right. We can understand $|X|$ in the formula as the true label of the image$|Y|$can be understood as the predicted results generated through the network; The $|X \cap Y|$on the denominator is the intersection of the real label and the predicted result, that is, the correctly predicted part of the segmentation result. Therefore, the Dice coefficient can be understood as:

(2 * Correctly predicted results)/(True label+predicted results)

(a) Original (b) Ground truth (c) pred

Segmenting fundus in OCTA images

Example of Segmenting Fundus in OCTA Images



(a) Original (b) Ground truth (c) pred

Segmenting fundus in OCTA images

The first image on the left is of the retinal superficial vascular complex, with the segmentation label in the middle and the segmentation prediction results generated by the neural network on the far right. We can understand $|X|$ in the formula as the true label of the image$|Y|$can be understood as the predicted results generated through the network; The $|X \cap Y|$on the denominator is the intersection of the real label and the predicted result, that is, the correctly predicted part of the segmentation result. Therefore, the Dice coefficient can be understood as:

$$\frac{2*\text{correctly predicted results}}{\text{true label+predicted results}}$$

The Calculation Process of Dice Coefficient

Assuming the segmentation labels $|X|$ of the image and the predicted results $|Y|$ obtained through the model are as follows:

$$|X| = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \qquad |Y| = \begin{bmatrix} 0.02 & 0.73 & 0.13 & 0.85 \\ 0.11 & 0.03 & 0.05 & 0.09 \\ 0.98 & 0.15 & 0.89 & 0.10 \\ 0.93 & 0.82 & 0.07 & 0.97 \end{bmatrix}$$

The dot product between the real label and the predicted result can be calculated as follows:

$$|X \cap Y| = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0.02 & 0.73 & 0.13 & 0.85 \\ 0.11 & 0.03 & 0.05 & 0.09 \\ 0.98 & 0.15 & 0.89 & 0.10 \\ 0.93 & 0.82 & 0.07 & 0.97 \end{bmatrix} = \begin{bmatrix} 0 & 0.73 & 0 & 0.85 \\ 0 & 0 & 0 & 0 \\ 0.98 & 0 & 0.89 & 0 \\ 0.93 & 0.82 & 0 & 0.97 \end{bmatrix}$$

The Calculation Process of Dice Coefficient

The Dice loss is calculated based on the Dice coefficient, and its calculation formula is as follows：

$$L = 1 - \frac{2|X \cap Y|}{|X| + |Y|}$$

Therefore, we can further calculate the Dice loss of the sample as:

$$L = 1 - \frac{2|X \cap Y|}{|X| + |Y|} = 1 - \frac{2\times 6.17}{13.92} \approx 0.1135$$

The larger the Dice coefficient, the more similar the set is, and the smaller the Dice loss; The reverse is also true. Dice loss is applicable to situations where the sample is extremely uneven.

The Calculation Process of Dice Coefficient

The sum of all elements in the result of element by element multiplication is:

$$|X \cap Y| = \begin{bmatrix} 0 & 0.73 & 0 & 0.85 \\ 0 & 0 & 0 & 0 \\ 0.98 & 0 & 0.89 & 0 \\ 0.93 & 0.82 & 0 & 0.97 \end{bmatrix}$$

The calculation of the denominator $|Y|+|Y|$ generally adopts the method of adding elements directly or summing the squares of elements. Here, we adopt the method of adding elements directly. And $|X| + |Y|$ is:

$$|X| = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} = 7 \qquad |Y| = \begin{bmatrix} 0.02 & 0.73 & 0.13 & 0.85 \\ 0.11 & 0.03 & 0.05 & 0.09 \\ 0.98 & 0.15 & 0.89 & 0.10 \\ 0.93 & 0.82 & 0.07 & 0.97 \end{bmatrix} = 6.92$$

$$|X| + |Y| = 7 + 6.92 = 13.92$$

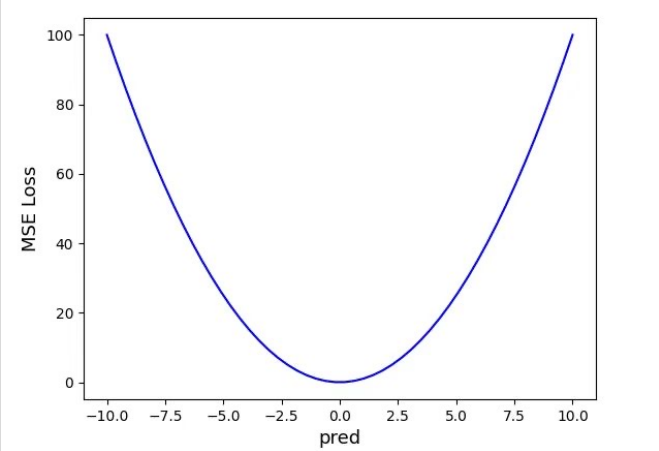
Regression: Mean Square Error Loss

The most commonly used loss function for regression problems is the mean squared error (MSE) loss function, which is defined as the sum of squares of the difference between the true value and the predicted value. Its form is as follows

$$MSE(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2,$$

where $y^{(i)}$ is the true value of the i-th sample, $\hat{y}^{(i)}$ is the predicted value of the i-th sample, and our optimization goal is to minimize the value of the loss function.

Regression: Mean Square Error Loss



Mean square error loss curve

Here, we further understand the characteristics of mean square error loss through a simple example. To simplify the discussion, assume that the sample size n=1 and the true value of the sample is 0. The following figure shows the predicted values $\hat{y} \in [-10,10]$. From this graph, it can be seen that the advantage of mean square error loss lies in its continuous and smooth curve points, making it easy to obtain derivatives. But its disadvantage is that it is more sensitive to outliers. When the distance between the predicted value and the true value is too far, the gradient obtained by using the gradient descent method may explode, leading to significant fluctuations in the training process.

Example of Taking the Regression Problem of Approximate Severity Prediction

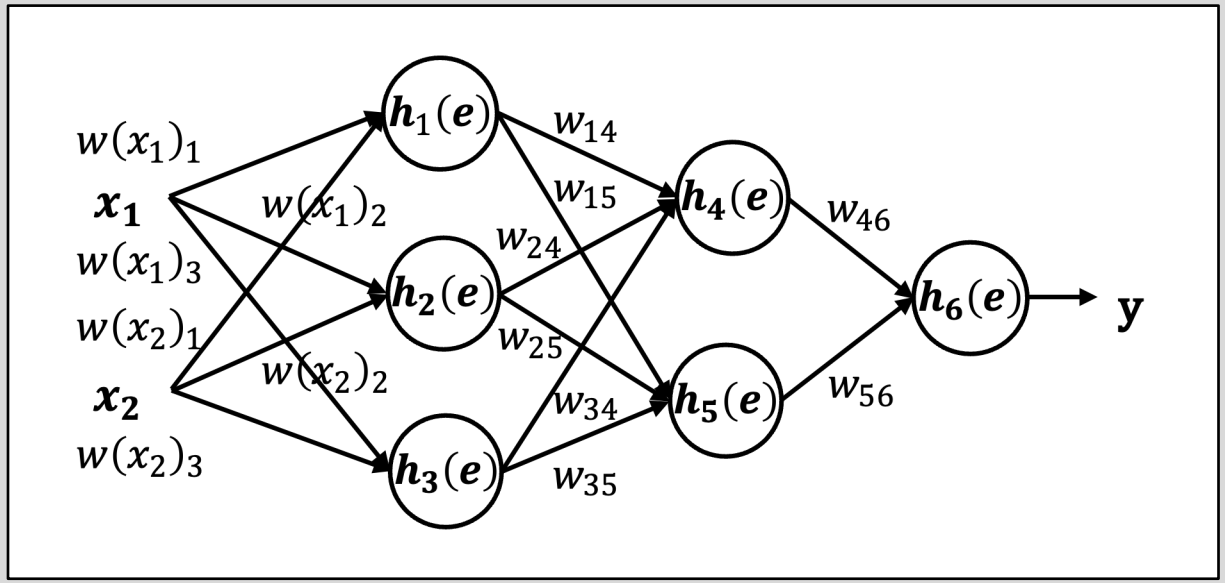
Mean square error function:

$$MSE(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2,$$

Assuming there are three samples with corresponding labels {9, 6, 8}, and the predicted values output by the model are {8.5, 6.7, 7.8}. According to the form of the mean square error loss function, the mean square loss can be calculated as：

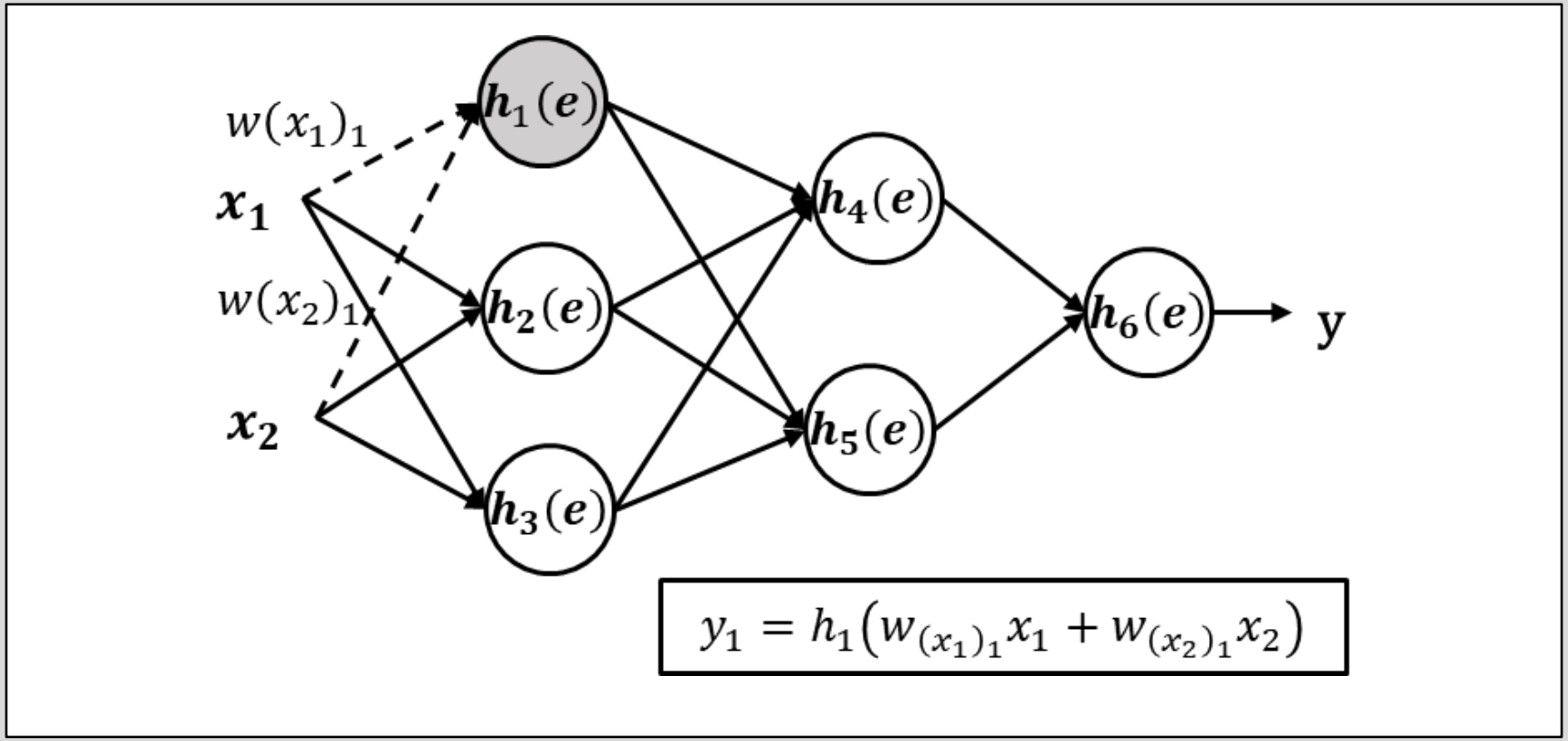
$$MSE(y, \hat{y}) = \frac{1}{3}[(9 - 8.5)^2 + (6 - 6.7)^2 + (8 - 7.8)^2] = 0.26$$

Neural Network Training Example

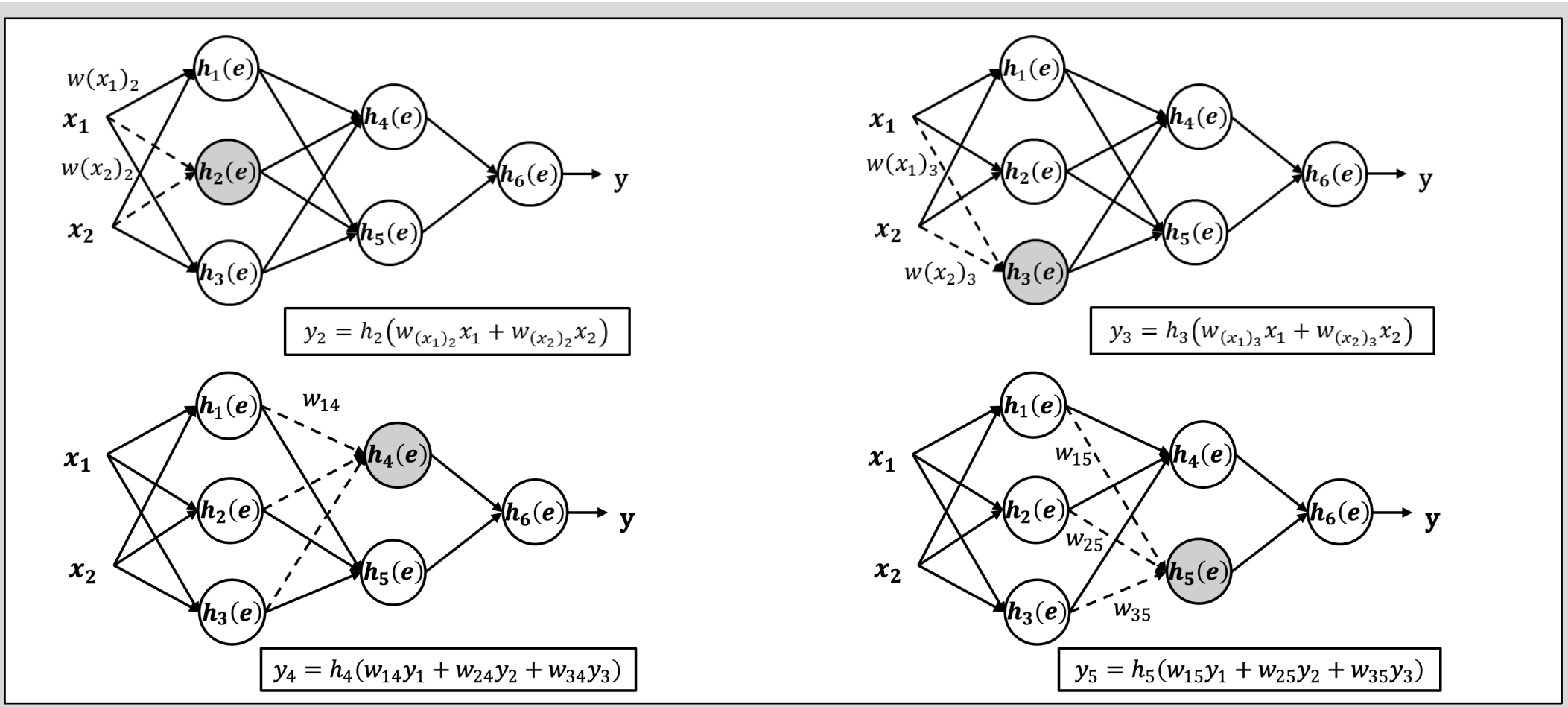


To obtain satisfactory parameters for a neural network, we typically need to train it using labeled data. The given dataset consists of input samples (x_1 and x_2) and their corresponding labels (expected output $\hat{y}$). Neural network training is an **iterative** process. In each iteration, the weights of the nodes are modified using new data from the dataset. Here, we use this multi-layer neural network as an example to explain the backpropagation algorithm for incentive propagation and weight updating.

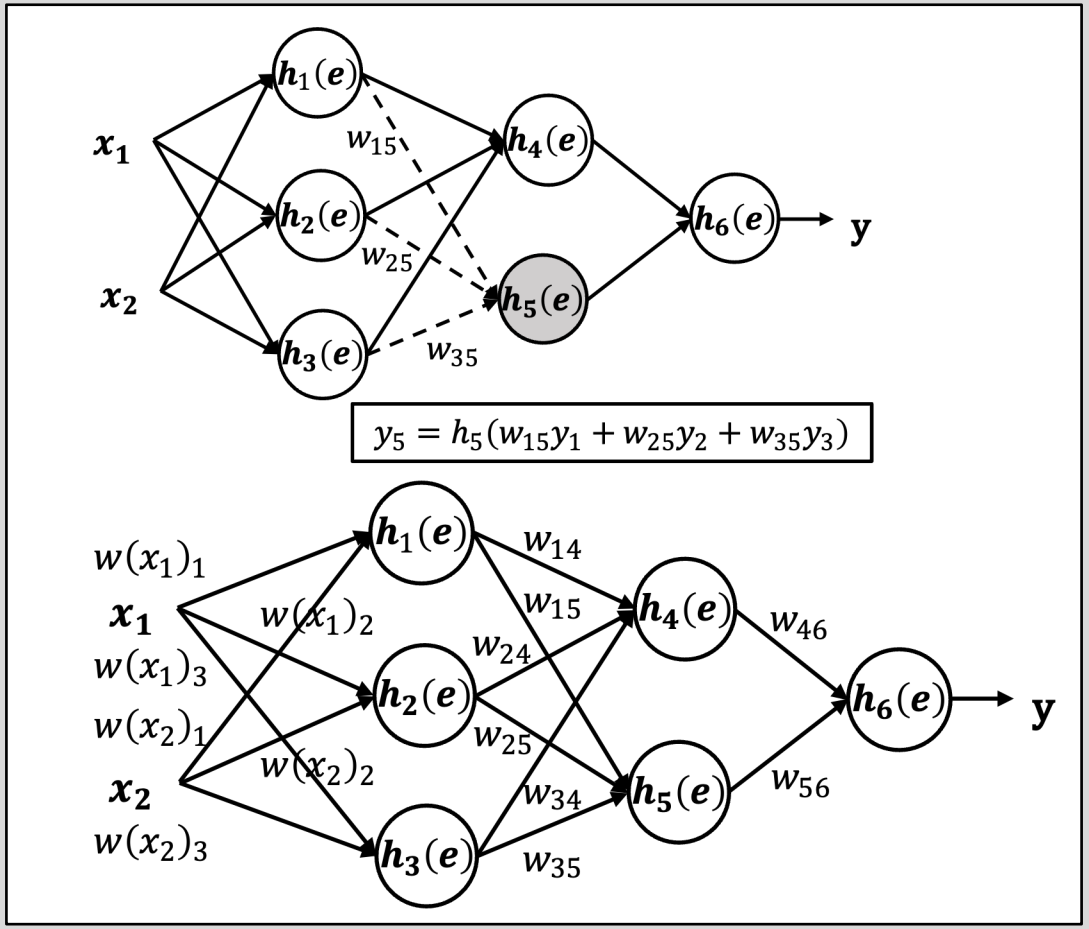
Forward Propagation



Forward Propagation



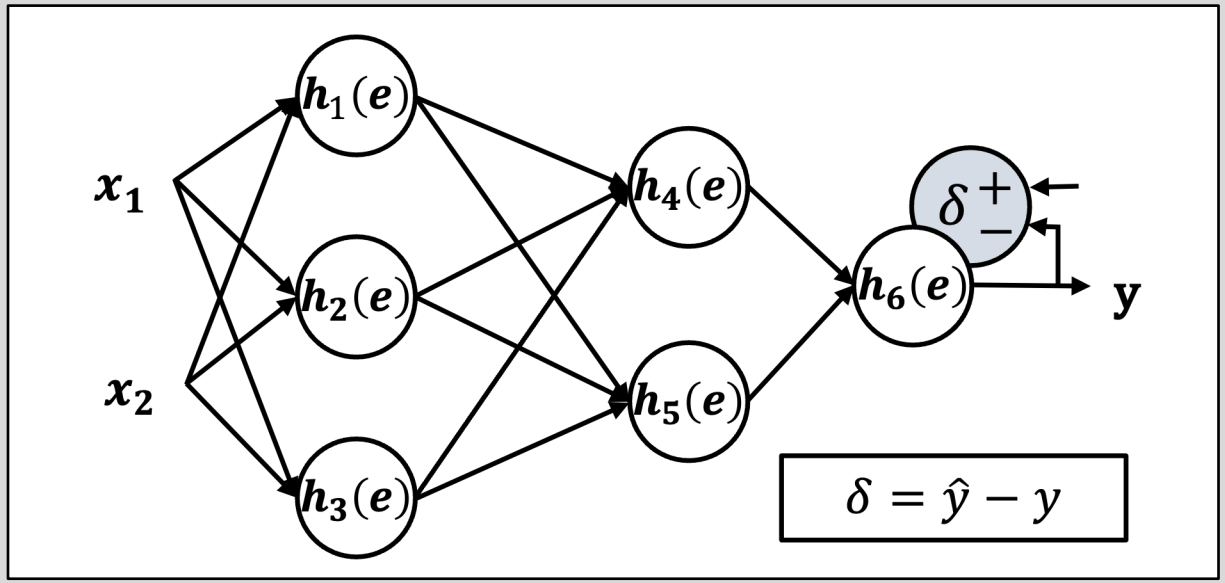
Forward Propagation



The symbol w_{mn} represents the connection weight between neuron m's output and neuron n's input in the next layer. As the signals pass through each neuron, the output signals are respectively:

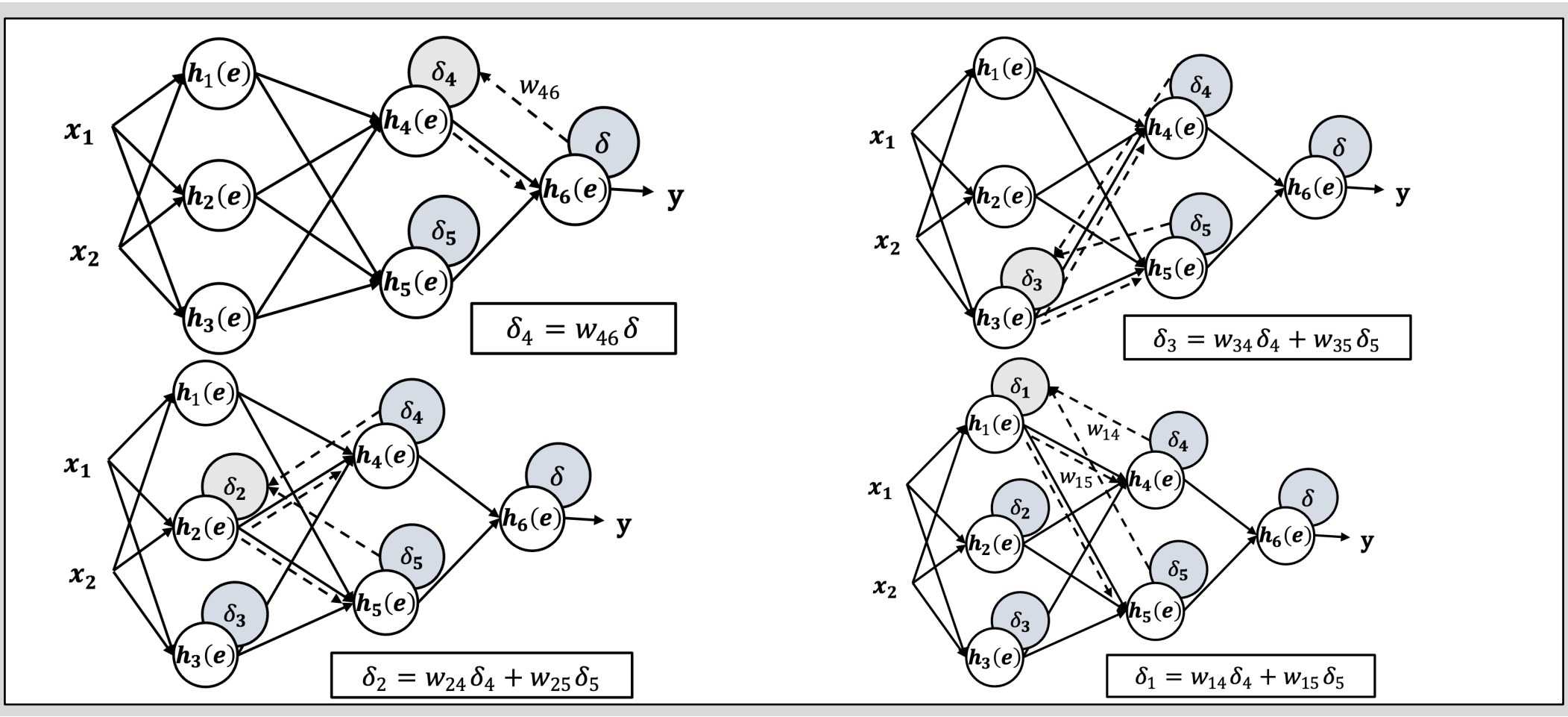
$$y_1 = h_1\left(w_{(x_1)_1}x_1 + w_{(x_2)_1}x_2\right)$$
$$y_2 = h_2\left(w_{(x_1)_2}x_1 + w_{(x_2)_2}x_2\right)$$
$$y_3 = h_3\left(w_{(x_1)_3}x_1 + w_{(x_2)_3}x_2\right)$$
$$y_4 = h_4(w_{14}y_1 + w_{24}y_2 + w_{34}y_3)$$
$$y_5 = h_5(w_{15}y_1 + w_{25}y_2 + w_{35}y_3)$$
$$y = h_6(w_{46}y_4 + w_{56}y_5)$$

Backward Propagation

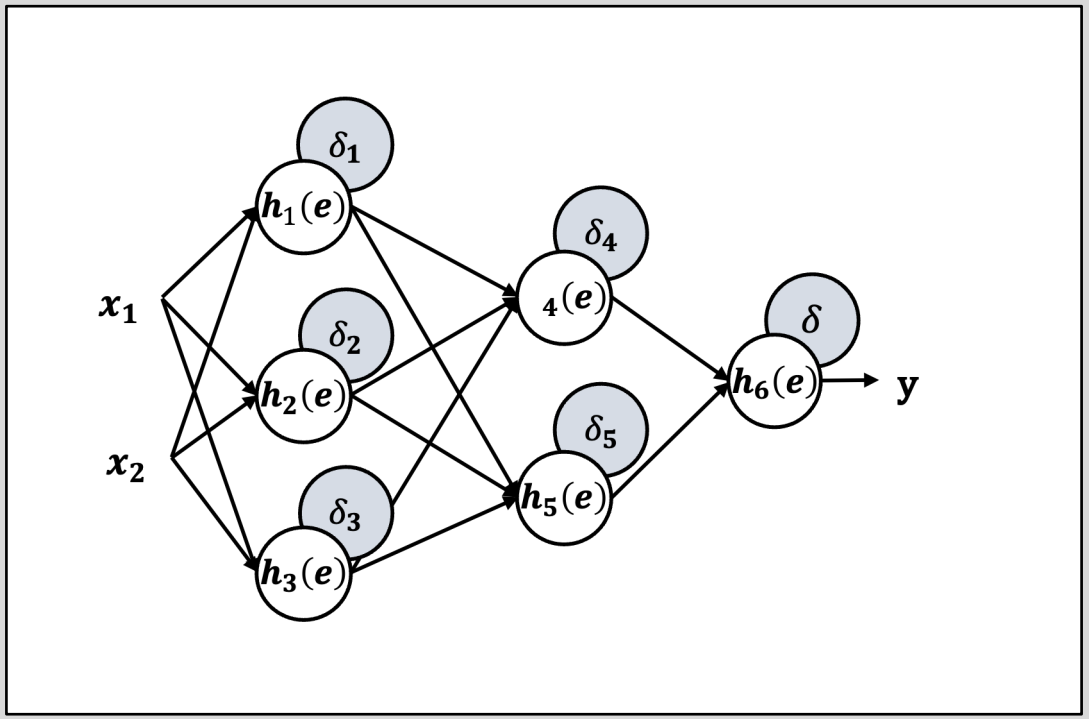


In backpropagation, gradients of intermediate variables and parameters of a neural network are computed sequentially in the order from the output layer to the input layer. First, the output signal of the network, denoted as y, is compared to the expected output values in the training dataset. The difference is referred to as the response error δ of the output layer neuron, i.e., δ = $\hat{y}$ - y.

Backward Propagation



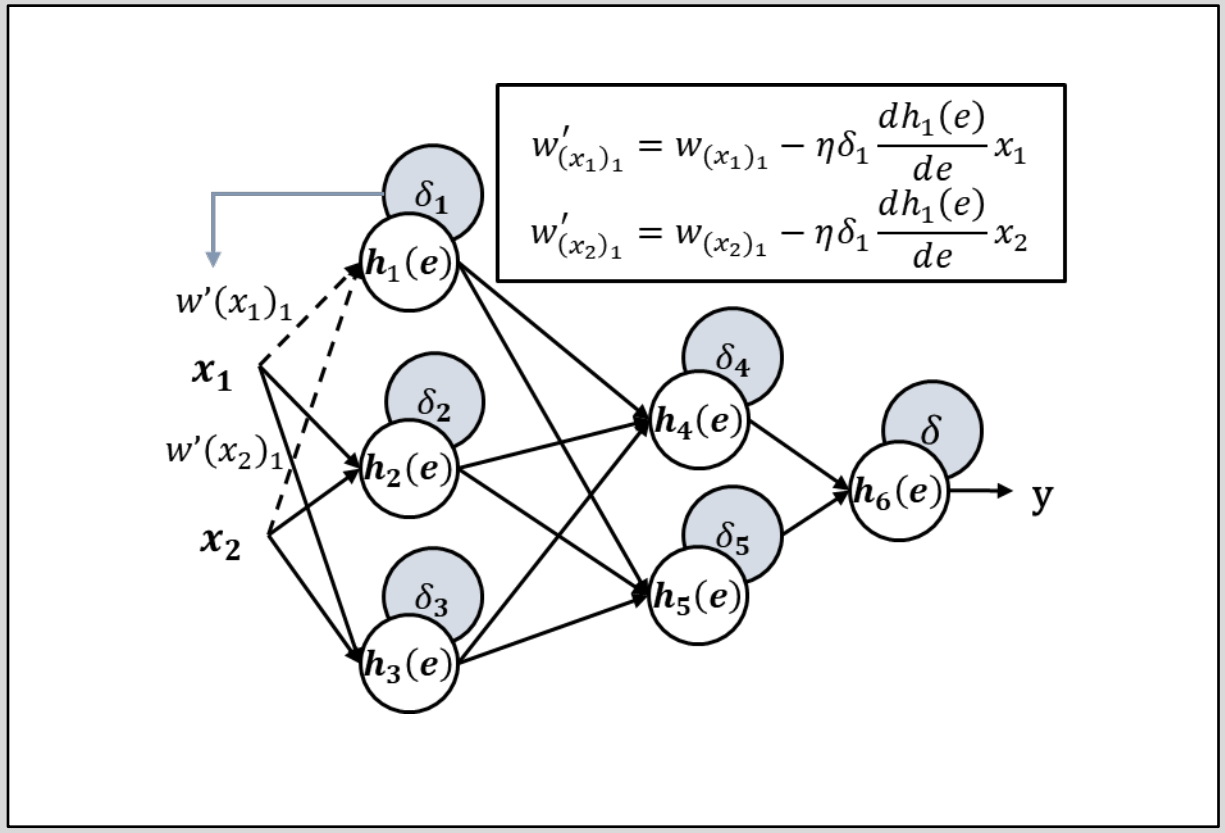
Backward Propagation



The symbol w_{mn} represents the connection weight between neuron m's output and neuron n's input in the next layer. As it passes through each neuron, the output signals are respectively:

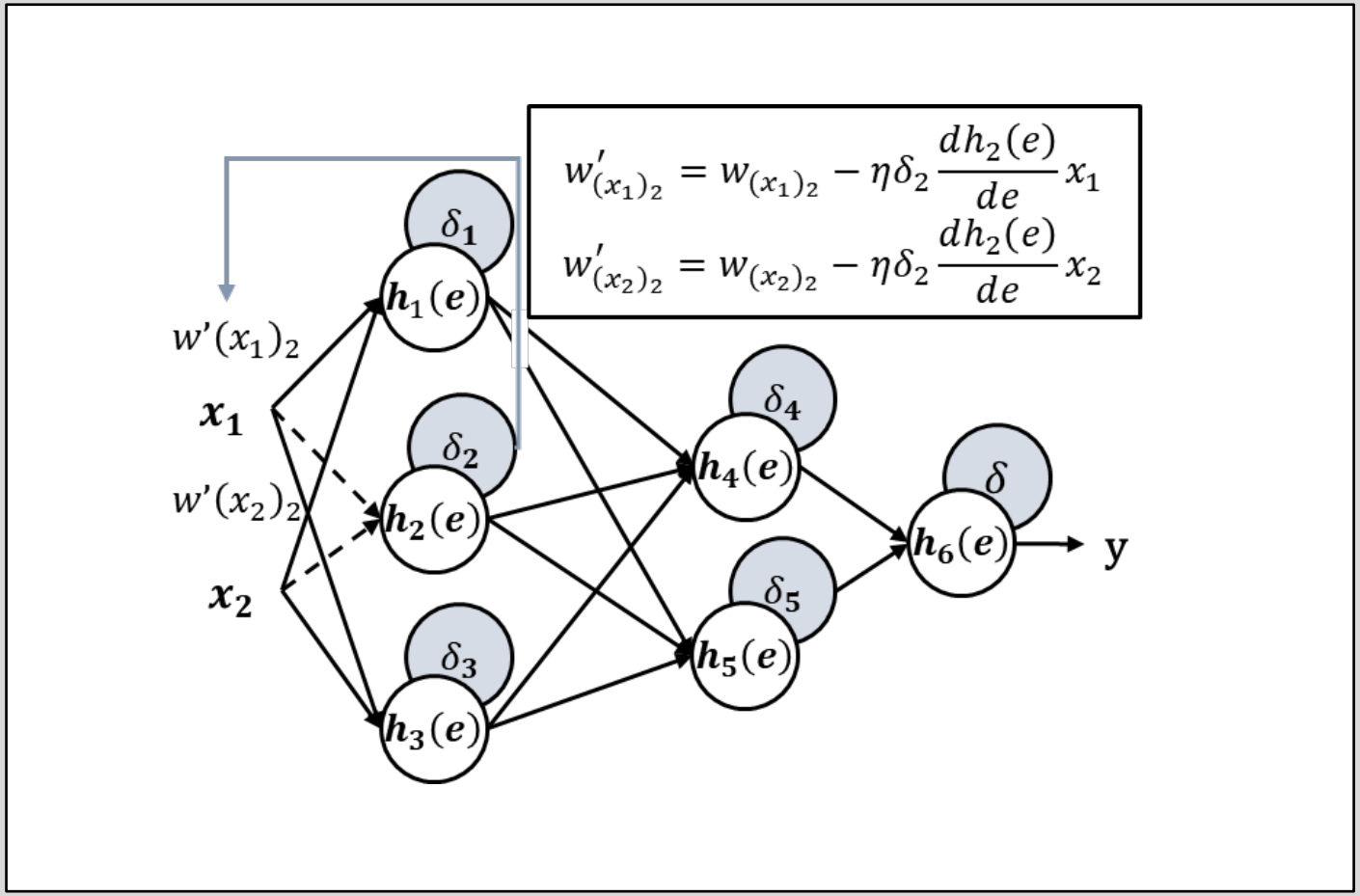
$$\delta_1 = w_{14}\delta_4 + w_{15}\delta_5$$
$$\delta_2 = w_{24}\delta_4 + w_{25}\delta_5$$
$$\delta_3 = w_{34}\delta_4 + w_{35}\delta_5$$
$$\delta_4 = w_{46}\delta$$
$$\delta_5 = w_{56}\delta$$

Weight Updating

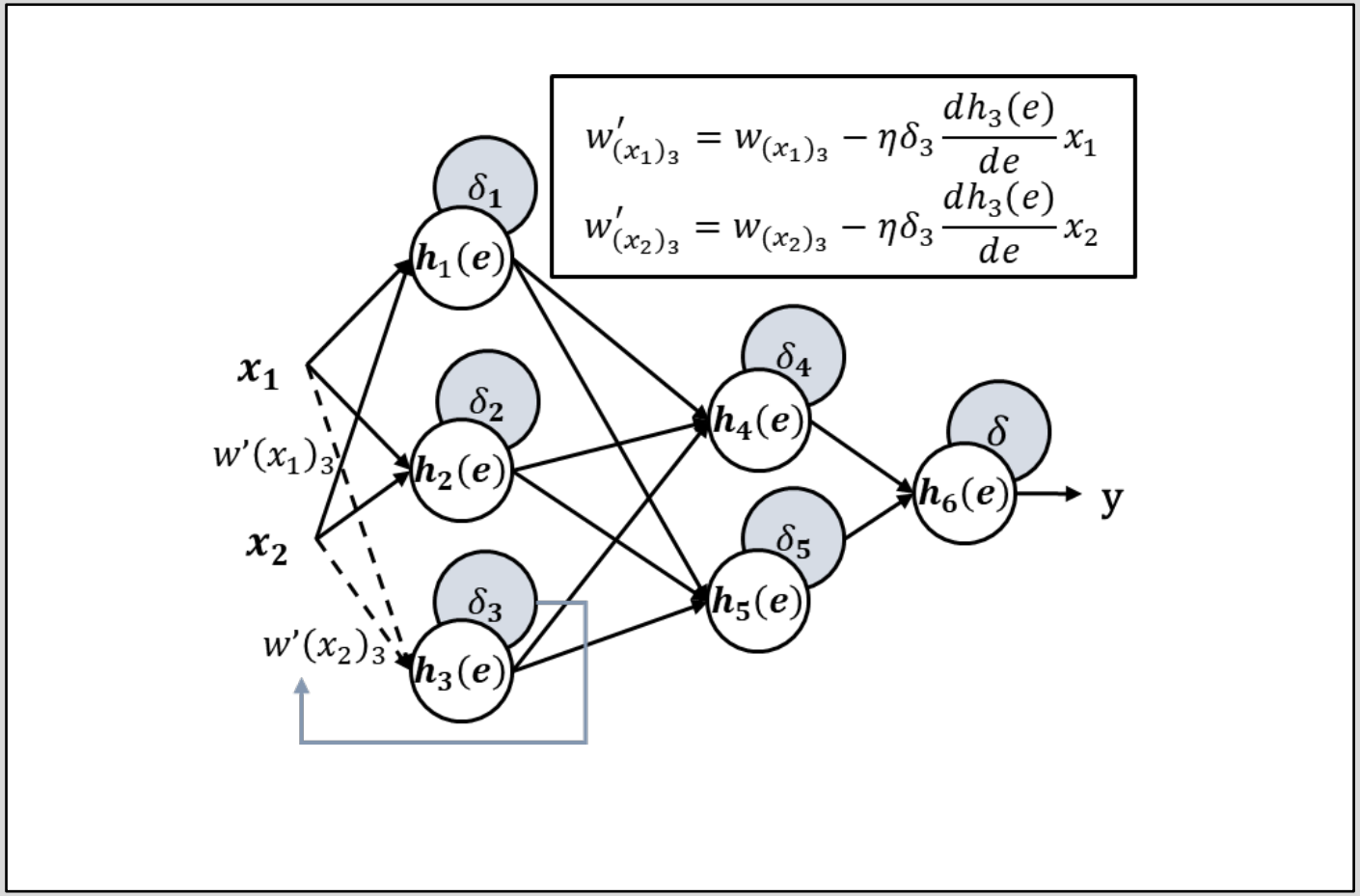


When computing the response error for each neuron, the weight coefficients of each neuron's input nodes are modified. In the following formula, $\frac{\mathrm{d}h(\mathrm{e})}{\mathrm{de}}$ represents the derivative of the neuron activation function, and the parameter η influences the learning rate of the network.

Weight Updating

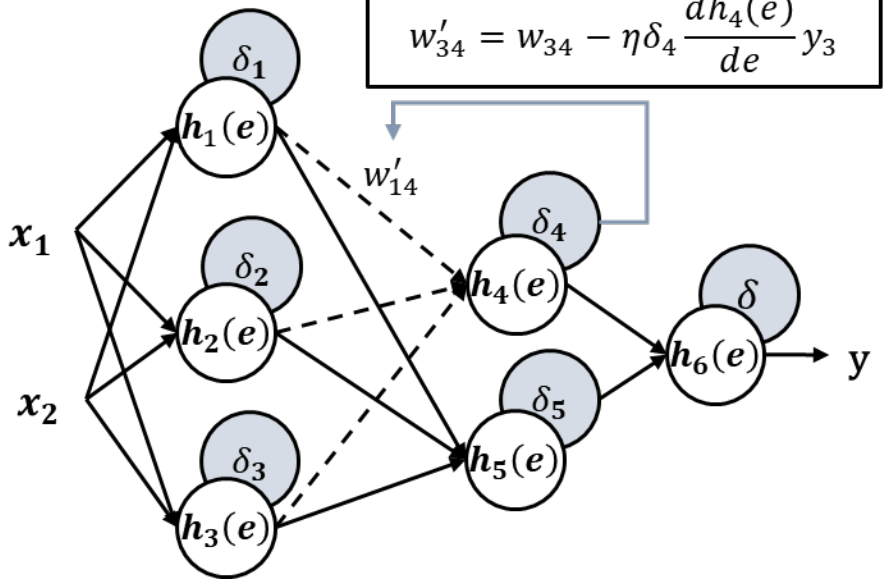


Weight Updating



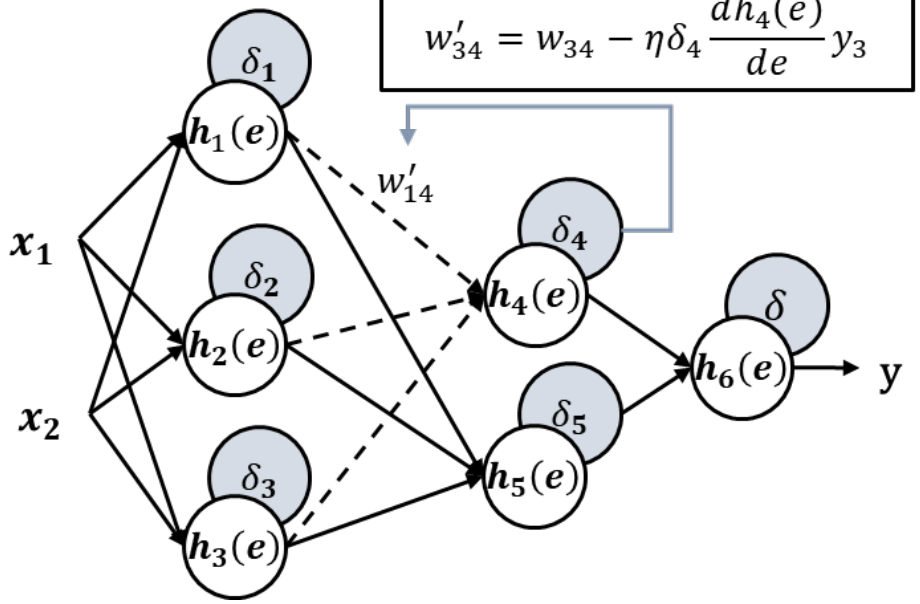
Weight Updating

$$w'_{14} = w_{14} - \eta\delta_4 \frac{dh_4(e)}{de} y_1$$
$$w'_{24} = w_{24} - \eta\delta_4 \frac{dh_4(e)}{de} y_2$$
$$w'_{34} = w_{34} - \eta\delta_4 \frac{dh_4(e)}{de} y_3$$

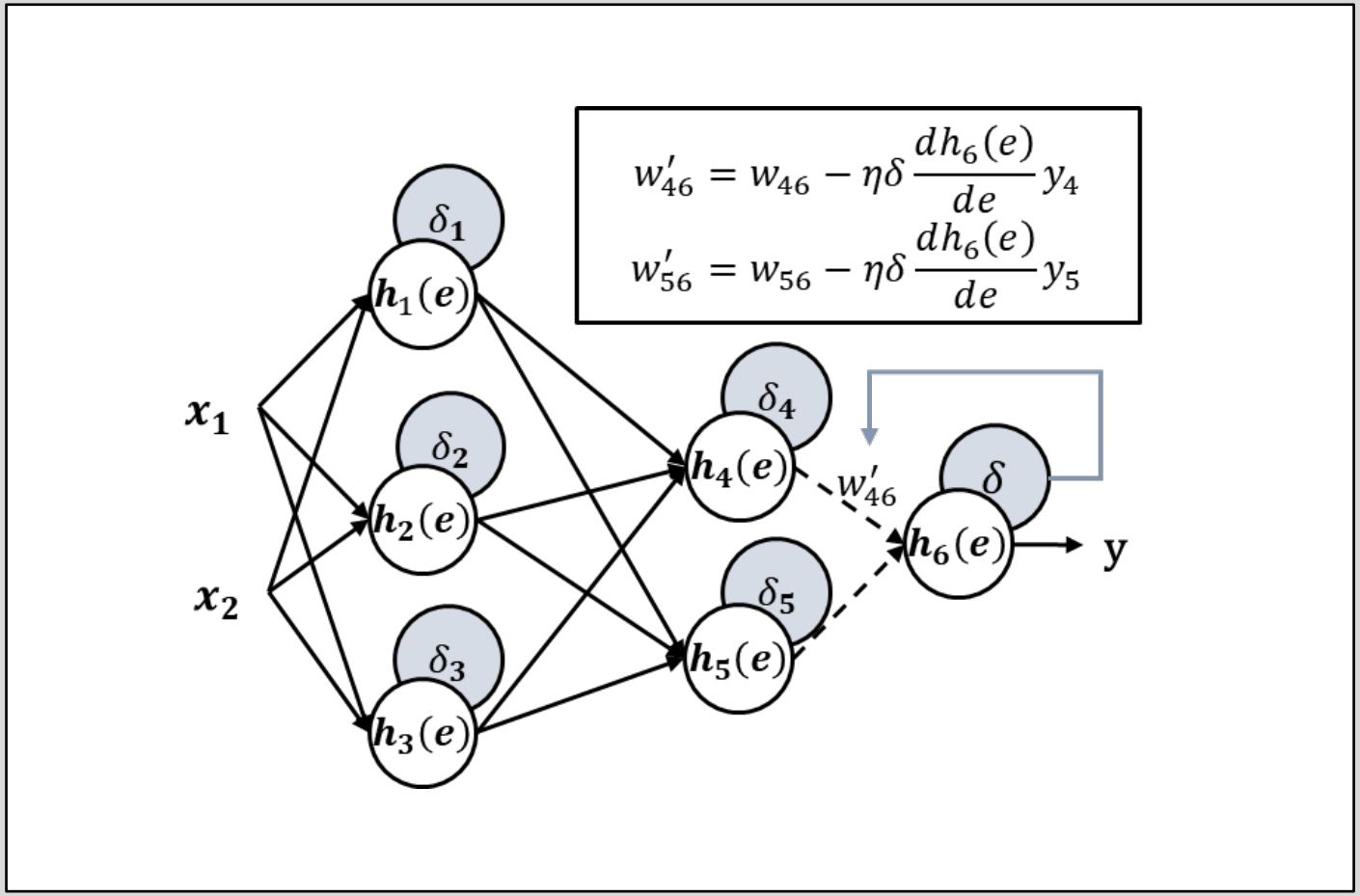


Weight Updating

$$w'_{14} = w_{14} - \eta\delta_4 \frac{dh_4(e)}{de} y_1$$
$$w'_{24} = w_{24} - \eta\delta_4 \frac{dh_4(e)}{de} y_2$$
$$w'_{34} = w_{34} - \eta\delta_4 \frac{dh_4(e)}{de} y_3$$



Weight Updating



Weight Updating

$$w'_{(x_1)_1} = w_{(x_1)_1} - \eta\delta_1 \frac{dh_1(e)}{de} x_1, \quad w'_{(x_2)_1} = w_{(x_2)_1} - \eta\delta_1 \frac{dh_1(e)}{de}$$

$$w'_{(x_1)_2} = w_{(x_1)_2} - \eta\delta_2 \frac{dh_2(e)}{de} x_1, \quad w'_{(x_2)_2} = w_{(x_2)_2} - \eta\delta_2 \frac{dh_2(e)}{de} x_2$$

$$w'_{(x_1)_3} = w_{(x_1)_3} - \eta\delta_3 \frac{dh_3(e)}{de} x_1, \quad w'_{(x_2)_3} = w_{(x_2)_3} - \eta\delta_3 \frac{dh_3(e)}{de} x_2$$

$$w'_{14} = w_{14} - \eta\delta_4 \frac{dh_4(e)}{de} y_1, \quad w'_{24} = w_{24} - \eta\delta_4 \frac{dh_4(e)}{de} y_2, \quad w'_{34} = w_{34} - \eta\delta_4 \frac{dh_4(e)}{de} y_3$$

$$w'_{15} = w_{15} - \eta\delta_5 \frac{dh_5(e)}{de} y_1, \quad w'_{25} = w_{25} - \eta\delta_5 \frac{dh_5(e)}{de} y_2, \quad w'_{35} = w_{35} - \eta\delta_5 \frac{dh_5(e)}{de} y_3$$

$$w'_{46} = w_{46} - \eta\delta \frac{dh_6(e)}{de} y_4, \quad w'_{56} = w_{56} - \eta\delta \frac{dh_6(e)}{de} y_5$$

Backpropagation Algorithm

Algorithm 2 Back Propagation (BP) Algorithm

Input: dataset $D = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$, learning rate η
Output: W
1: randomly initialize W
2: **repeat**
3: reorder the samples in dataset D
4: **for** $n = 1 \ldots$ N **do**
5: select sample $(x^{(n)}, y^{(n)})$ from the dataset D
6: calculate the output of each layer forward until the last layer
7: back propagation calculates the error of each layer and updates the parameters
8: **end**
9: **until** the neural network reaches a certain termination condition

Homework 12-1

- The following table shows the sample data of 3 cases of myopia. The labels are {0,1,2}, where 0 represents normal samples, 1 represents mild myopia, and 2 represents severe myopia. The table shows the results of the model predicting samples into various classes. Please write the one-hot encoding corresponding to the true label of the sample based on the data in the table and calculate the cross-entropy loss.

Number	Normal	Mild Myopia	Severe Myopia
1	0.6	0.3	0.1
2	0.2	0.5	0.3
3	0.0	0.25	0.75

Homework 12-2

Assume that the segmentation label $|X|$ of the image and the prediction result $|Y|$ obtained by the model are as follows:

$$|X| = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \qquad |Y| = \begin{bmatrix} 0.05 & 0.83 & 0.13 & 0.85 \\ 0.15 & 0.03 & 0.05 & 0.19 \\ 0.77 & 0.21 & 0.91 & 0.10 \\ 0.94 & 0.82 & 0.07 & 0.97 \end{bmatrix}$$

Please calculate the Dice coefficient according to the method of summing the squares of elements:

Reference formula: $$Dice(X,Y) = \frac{2\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} y_i^2}$$

Homework 12-3

Assume there are 4 samples,
The labels corresponding to the samples are {3, -1, 2, -5},
The predicted values output by the model are {2.5, -1.2, 3.1, -4.9}.
Please calculate the mean square error loss.

Introduction of AI (CS103)- 12 Deep Learning - BP

Jimmy Liu 刘江

2023-12-08